ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

MEMOIRE DE PROJET DE FIN D'année

Filière

Génie Informatique et Réseaux

Elaboré par

BARDICH HAMZA ET AIT MBAREK OTHMANE

Sous le thème

Traducteur de langage des signes

Soutenu en session de mai 2025 sous la direction de :

Mme. Amina Ouatiq — Encadrante Pédagogique

Année universitaire 2024 - 2025

Remerciements

Nous tenions à remercier de prime abord Dieu pour sa bonté et sa gratitude sans lesquelles nous ne pourrions en aucun cas être ici.

Nous tenons à exprimer nos sincères remerciements à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet.

Nous remercions tout particulièrement notre encadrante, madame Ouatiq Amina, pour sa disponibilité, ses conseils précieux et son accompagnement tout au long de ce travail. Son expertise et ses orientations nous ont été d'une grande aide pour mener à bien ce projet.

Nous adressons également notre gratitude à l'ensemble de l'équipe pédagogique ainsi qu'à toutes les personnes ayant facilité le bon déroulement de notre travail.

Résumé

Ce projet présente le développement d'un système de traduction de la langue des signes en texte, utilisant la vision par ordinateur et l'intelligence artificielle. Le but est de reconnaître les lettres de l'alphabet signées avec la main et de les afficher automatiquement à l'écran.

Le fonctionnement du système repose sur l'activation de la caméra, qui capte en temps réel les gestes effectués par l'utilisateur. Chaque signe représentant une lettre est analysé par un modèle d'apprentissage automatique, qui identifie la lettre correspondante. En plus d'afficher la lettre reconnue, le système fournit un pourcentage de confiance, indiquant la probabilité que la prédiction soit correcte.

Ce projet vise à faciliter la communication avec les personnes sourdes et muettes en traduisant visuellement les signes en texte. Il utilise des technologies comme Python, OpenCV, MediaPipe , tenserflow , Keras et des bibliotheque de python.

Introduction générale

L'intelligence artificielle (IA) est l'un des domaines les plus révolutionnaires de la technologie moderne. Elle vise à développer des machines et des systèmes capables de réaliser des tâches qui nécessitent généralement de l'intelligence humaine, comme la reconnaissance d'images, la prise de décision, ou encore l'apprentissage de nouvelles compétences. Grâce à des algorithmes sophistiqués et à l'analyse de grandes quantités de données, l'IA permet de résoudre des problèmes complexes plus rapidement et avec plus de précision que les méthodes traditionnelles. Cette technologie trouve aujourd'hui des applications dans divers secteurs, tels que la médecine, les transports, l'industrie etc…

Parmi les différentes branches de l'intelligence artificielle, le deep learning (apprentissage profond) se distingue comme l'une des plus avancées et puissantes. Le deep learning repose sur des réseaux de neurones artificiels à plusieurs couches, qui sont inspirés du fonctionnement du cerveau humain. Ces réseaux sont capables d'analyser des données massives et complexes, d'en extraire des motifs et des caractéristiques pertinentes, et de réaliser des prédictions ou des classifications avec une grande précision. Contrairement aux approches classiques de l'apprentissage automatique, le deep learning ne nécessite pas de processus manuel pour sélectionner les caractéristiques les plus importantes dans les données. Il apprend de manière autonome à partir des données brutes, ce qui le rend particulièrement efficace dans des domaines comme la vision par ordinateur, le traitement du langage naturel, ou encore la reconnaissance vocale.

La langue des signes constitue un moyen de communication essentiel pour les personnes sourdes et malentendantes. Toutefois, son usage reste limité dans les échanges avec les personnes ne la maîtrisant pas. Ces dernières années, les avancées en **intelligence artificielle**, et plus particulièrement en **deep learning**, ont ouvert de nouvelles perspectives pour le développement de systèmes capables d’interpréter automatiquement les gestes de la langue des signes.

Le **deep learning**, grâce à ses réseaux de neurones profonds, permet d’analyser efficacement des images et des séquences vidéo afin de reconnaître des formes complexes, comme les mouvements et les positions des mains. Appliqué à la reconnaissance gestuelle, il permet de créer des modèles capables d’identifier les signes correspondant aux lettres ou aux mots, avec un haut niveau de précision. Ces technologies rendent possible la traduction en temps réel des gestes captés par une caméra en texte lisible, facilitant ainsi l’inclusion et la communication entre les communautés.

Abstract

Ce projet se concentre sur le développement d'un traducteur de langue des signes en temps réel, reposant sur les techniques de deep learning et de vision par ordinateur. À l'aide d'une caméra, le système capture les gestes de la main représentant des lettres, reconnaît chaque signe et affiche la lettre correspondante à l'écran, accompagnée d'un pourcentage de confiance indiquant la précision de la reconnaissance. Cette technologie vise à combler les lacunes de communication entre les personnes sourdes ou malentendantes et celles qui ne connaissent pas la langue des signes, en offrant un outil de traduction accessible et instantané.

L'intelligence artificielle (IA) est aujourd'hui l'un des domaines les plus transformateurs de la technologie moderne. Son objectif est de créer des machines et des systèmes capables d'accomplir des tâches qui nécessitent généralement l'intelligence humaine, telles que la reconnaissance d'images, la prise de décision ou l'apprentissage de nouvelles compétences. En exploitant des algorithmes avancés et l'analyse de grandes quantités de données, l'IA permet de résoudre des problèmes complexes plus rapidement et avec plus de précision que les méthodes traditionnelles. Aujourd'hui, ses applications couvrent un large éventail de secteurs, notamment la santé, les transports et l'industrie.

Parmi les différentes branches de l'IA, le deep learning s'impose comme l'une des techniques les plus puissantes et sophistiquées. Il repose sur des réseaux neuronaux artificiels multicouches, inspirés de l'architecture du cerveau humain. Ces réseaux excellent dans le traitement de données vastes et complexes, l'extraction de motifs et de caractéristiques significatives, et la production de prédictions ou de classifications très précises. Contrairement aux méthodes traditionnelles d'apprentissage automatique qui nécessitent une sélection manuelle des caractéristiques, le deep learning apprend de manière autonome directement à partir des données brutes. Cette capacité le rend particulièrement efficace dans des domaines comme la vision par ordinateur, le traitement du langage naturel et la reconnaissance vocale.

La langue des signes est un outil de communication essentiel pour les personnes sourdes et malentendantes. Cependant, son usage reste limité dans les échanges avec ceux qui ne la maîtrisent pas. Les avancées récentes de l'IA, et en particulier du deep learning, ont ouvert la voie à des systèmes capables d'interpréter automatiquement les gestes de la langue des signes. En utilisant des réseaux neuronaux profonds, ces systèmes peuvent analyser des images ou des séquences vidéo afin de reconnaître avec précision les mouvements et positions complexes des mains. Appliquée à la reconnaissance gestuelle, cette technologie permet d'identifier les signes correspondant à des lettres ou à des mots avec une grande fiabilité. Ces innovations facilitent la traduction en temps réel des gestes capturés par une caméra.

Table des matières

Liste des Figures

Chapitre I

Cahier des charges

1. Présentation du projet

A. Contexte du projet :

La langue des signes est un moyen de communication visuel et gestuel essentiel pour les personnes sourdes ou malentendantes. Malgré son importance, elle reste peu comprise par le grand public, ce qui crée des barrières sociales, professionnelles et éducatives. Les technologies modernes d'intelligence artificielle et de vision par ordinateur offrent aujourd'hui la possibilité de développer des outils capables de traduire automatiquement les signes en texte ou en parole, facilitant ainsi l'inclusion et l'accessibilité.

Ce projet vise à créer un traducteur de langue des signes (ASL - American Sign Language) en temps réel, en s'appuyant sur des bibliothèques open-source telles que OpenCV, MediaPipe, TensorFlow et Teachable Machine**.** L'objectif est de fournir une solution précise, rapide et facile à utiliser

B. Problématique du projet :

Malgré les efforts d'inclusion, les personnes sourdes et malentendantes rencontrent toujours des obstacles dans leur communication quotidienne avec les personnes entendantes. La langue des signes, bien qu'essentielle pour cette communauté, n'est pas maîtrisée par la majorité de la population. Cette barrière linguistique limite l'accès à l'éducation, à l'emploi, aux services publics et à la vie sociale. Il devient donc crucial de développer un système capable de **traduire automatiquement la langue des signes en texte**, afin de faciliter l'interaction entre sourds et entendants, tout en favorisant l'autonomie et l'inclusion

C. Objectifs de projet

Le système devra remplir les fonctions suivantes :

Détection et analyse des gestes :

- Utilisation de MediaPipe pour extraire les points clés des mains
- Reconnaissance des 26 lettres de l'alphabet (A-Z)
- Précision minimale de 90% en conditions optimales
- Détection en temps réel

Reconnaissance des signes :

Classification des gestes via un modèle entraîné avec **Teachable Machine** et optimisé avec **TensorFlow**

Traduction en temps réel :

Affichage immédiat du texte correspondant au signe détecté.

Interface conviviale :

- Application simple et intuitive
- Flux vidéo de la webcam
- Lettre reconnue (texte gros format)
- Indicateur de confiance (%)

D. Public cible et application :

- **Personnes sourdes ou malentendantes :** Pour communiquer plus facilement avec les entendants.
- **Établissements scolaires et professionnels** : Pour favoriser l'inclusion en milieu éducatif ou en entreprise.
- **Développeurs et chercheurs :** Comme base open-source pour des projets plus avancés.

Besoins du projet :

1. **Fonctionnel** :
    - Reconnaissance fiable des 26 lettres de l'alphabet ASL
    - Affichage texte immédiat et lisible
2. **Technique** :
    - Latence minimale
    - Compatibilité avec les webcams standard
    - Faible consommation des ressources
3. **Utilisateur** :
    - Interface intuitive sans apprentissage préalable
    - Accessibilité visuelle (taille des caractères, contrastes)
    - Mode démo intégré

2. Conclusion :

Le développement d'un système de traduction de la langue des signes en texte représente une avancée technologique importante pour l'inclusion sociale des personnes sourdes et malentendantes. Ce projet vise à réduire les barrières de communication en proposant une solution accessible, intelligente et en temps réel. Grâce à l'intégration de technologies modernes telles que la vision par ordinateur et l'intelligence artificielle, il devient possible d'automatiser partiellement la compréhension de la langue des signes, facilitant ainsi les échanges avec les personnes entendantes.
Ce projet, bien que limité dans sa première version à un ensemble restreint de signes, ouvre la voie à des évolutions futures telles que la reconnaissance de phrases complètes, la synthèse vocale ou encore la traduction bidirectionnelle (texte vers langue des signes animée). Il s'inscrit pleinement dans une démarche inclusive, innovante et citoyenne.

Chapitre II

Architecture du système

Introduction :

Ce chapitre présente l'architecture logicielle mise en place pour le développement du système de traduction de la langue des signes en texte. Il décrit les outils, bibliothèques et technologies qui ont été utilisés pour concevoir et implémenter les différentes fonctionnalités du projet. Le choix de ces composants s'est fait en fonction de leur performance, de leur simplicité d'intégration et de leur pertinence par rapport aux besoins du système, notamment en ce qui concerne la détection des gestes, le traitement des images et la reconnaissance des signes.

1. Python :

A. Présentation générale

Python est un langage de programmation interprété, open source, connu pour sa simplicité de syntaxe, sa lisibilité et sa grande communauté. Il est aujourd'hui largement utilisé dans de nombreux domaines tels que l'intelligence artificielle, le développement web, le traitement d'image, les sciences des données, l'automatisation et bien d'autres.

Créé à la fin des années 1980 par **Guido van Rossum**, Python est devenu l'un des langages les plus populaires grâce à sa capacité à intégrer de puissantes bibliothèques et à accélérer le prototypage des projets complexes.

B. Pourquoi Python pour ce projet ?

Le choix de Python pour ce système de traduction de la langue des signes repose sur plusieurs avantages majeurs :

- **Compatibilité avec les bibliothèques d'IA** comme TensorFlow, MediaPipe, NumPy, etc.
- **Facilité de traitement d'images** avec OpenCV.

- **Communauté très active** et documentation riche.
- **Rapidité de développement** grâce à une syntaxe simple et intuitive.
- **Multiplateforme** : fonctionne sous Windows, Linux et Mac sans modification du code.

C. Rôle de Python dans le projet

Dans notre projet, Python a permis de :

- Gérer le flux vidéo en temps réel à partir de la webcam.
- Intégrer et utiliser les modules MediaPipe pour la détection des mains.
- Traiter les données avec NumPy et math.
- Charger et exécuter le modèle d'intelligence artificielle exporté depuis Teachable Machine avec TensorFlow.
- Créer une interface de détection et d'affichage conviviale en ligne de commande.

D. Conclusion

Python a été le socle du développement de notre système de traduction de la langue des signes. Grâce à sa richesse en bibliothèques spécialisées et à sa facilité d'intégration avec des outils d'IA et de vision par ordinateur, il a permis de construire un pipeline complet, de l'acquisition vidéo à la reconnaissance intelligente des gestes, tout en conservant un code clair et modulaire.

2. Opencv :

A. Présentation générale

OpenCV ou (Open Source Computer Vision Library) est une bibliothèque open source dédiée au traitement d'images et à la vision par ordinateur. Développée initialement par Intel et désormais maintenue par une large communauté, elle est l'une des bibliothèques les plus populaires dans le domaine de la vision artificielle. Compatible avec plusieurs langages (Python, C++, Java), OpenCV permet de développer des applications capables d'analyser, de traiter et d'interpréter des images ou des flux vidéo.

B. Pourquoi OpenCV dans ce projet ?

Dans le cadre de notre système de traduction de la langue des signes, OpenCV a été utilisé principalement pour :

- **Capturer le flux vidéo** en temps réel depuis la webcam ;
- **Afficher ce flux** à l'utilisateur pour suivre la détection des gestes ;
- **Prétraiter les images** (par exemple : redimensionnement, conversion en niveaux de gris ou en RGB si besoin) ;
- Coordonner les différentes étapes du traitement visuel avant l'analyse par MediaPipe ou le modèle d'apprentissage automatique.

C. Avantages d'OpenCV

- **Compatibilité multiplateforme** : fonctionne sous Windows, Linux et macOS.
- **Documentation abondante** : nombreux tutoriels et exemples en ligne.
- **Performance** : traitement rapide des images en temps réel.
- **Interopérabilité** : s'intègre facilement avec d'autres bibliothèques comme MediaPipe, NumPy ou TensorFlow.

D. Limites et précautions

Même si OpenCV est puissant, il présente certaines limites :

- Il ne réalise aucune reconnaissance en soi : il faut le combiner avec d'autres bibliothèques (MediaPipe, modèles ML).
- Il peut être sensible à la qualité de la caméra, à l'éclairage ou aux bruits visuels.
- **La gestion des performances** (RAM/CPU) devient critique si le traitement est lourd ou mal optimisé.

E. Conclusion

OpenCV est un outil indispensable dans notre architecture logicielle. Il constitue la base du système de capture et d'affichage des données visuelles, en facilitant l'interaction entre l'utilisateur et le modèle de reconnaissance. Sa robustesse, sa polyvalence et sa large adoption dans le monde du traitement d'image en font un choix parfaitement adapté à un projet de traduction gestuelle en temps réel.

3. Mediapipe :

A. Présentation générale

MediaPipe est une bibliothèque open source développée par **Google** qui permet de construire des pipelines multimédias utilisant des modèles d'intelligence artificielle pour le suivi de mouvement, la détection de pose, la reconnaissance faciale, la détection des mains, etc.

Ce qui rend MediaPipe unique, c'est sa capacité à fournir des résultats précis en temps réel tout en étant léger, donc utilisable aussi bien sur des ordinateurs que sur des smartphones.

Dans notre projet de traduction de la langue des signes, MediaPipe a été utilisé pour détecter et suivre la position des mains et des doigts à partir d'un flux vidéo en direct.

B. Rôle de MediaPipe dans notre projet

MediaPipe est le cœur du système de détection gestuelle. Il permet d'obtenir les coordonnées exactes des points clés (landmarks) de la main, essentiels pour identifier les signes.

Ses rôles dans notre système sont :

- **Détecter automatiquement les mains** dans l'image sans entraînement préalable;
- **Extraire les coordonnées 3D** (x, y, z de la main (doigts, paume) ;
- Fournir des données structurées prêtes à être analysées ou utilisées dans un modèle de classification.

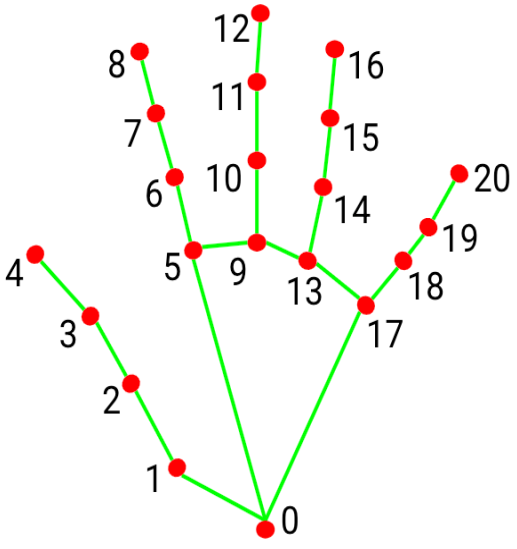
C. Avantages de MediaPipe

- **Très précis** même avec une seule caméra classique (pas besoin de capteurs 3D).
- **Temps réel** : détecte les mains à chaque image du flux vidéo.
- **Léger et rapide** : peut tourner sans GPU, même sur des machines modestes.
- **Facile à intégrer** avec OpenCV, NumPy et TensorFlow.

D. Structure des données retournées

MediaPipe retourne pour chaque main détectée :

- Une liste de **21 landmarks**.
- Chaque landmark contient :
  - `x` et `y` : position normalisée (entre 0 et 1) dans l'image.
  - `z` : profondeur relative.
  - Un identifiant fixe pour chaque point (ex. : `WRIST`, `THUMB_TIP`, `INDEX_FINGER_MCP`, etc.)



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 1: Mediapipe hand landmarks list

Ces données sont ensuite converties en vecteurs et utilisées comme entrée du modèle TensorFlow pour reconnaître le geste.

E. Limites de MediaPipe

- Ne détecte correctement les mains que si elles sont bien visibles (éclairage, orientation, taille dans l'image).
- Peut détecter plusieurs mains, mais cela demande une gestion supplémentaire.
- Ne fournit aucune classification des gestes : il faut utiliser un modèle pour cela (ex. : Teachable Machine + TensorFlow).

F. Conclusion

MediaPipe est un outil essentiel dans notre projet, car il automatise totalement la détection des mains et des doigts, avec une précision et une rapidité impressionnante. Il constitue le socle de l'entrée visuelle du système, rendant possible l'analyse gestuelle en temps réel sans capteurs physiques ni marqueurs.

4. Tensorflow :

A. Présentation générale

TensorFlow est une bibliothèque open source développée par **Google Brain**, conçue pour le développement et l'exécution de modèles d'apprentissage automatique et d'apprentissage profond. Elle permet de construire, entraîner et déployer des réseaux de neurones, qu'il s'agisse de simples classificateurs ou de modèles complexes de traitement d'images, de texte ou de sons.

TensorFlow est utilisé dans des domaines variés : reconnaissance d'image, traitement du langage naturel, prévision de séries temporelles, traduction automatique, etc. Il est particulièrement bien adapté aux projets d'intelligence artificielle embarquée ou en temps réel, ce qui en fait un choix stratégique pour notre système de traduction de la langue des signes.

B. Rôle de TensorFlow dans notre projet

Dans ce projet, TensorFlow est utilisé pour charger et exécuter un modèle de reconnaissance de gestes entraîné via Teachable Machine, également basé sur TensorFlow. Ce modèle prend en entrée les coordonnées des points clés détectés sur les mains (landmarks) et produit en sortie une prédiction du geste correspondant.

Les rôles de TensorFlow dans notre système sont donc les suivants :

- **Chargement du modèle pré-entraîné** ;
- **Prétraitement des données d'entrée** (normalisation des coordonnées) ;
- **Passage des données au modèle pour prédiction** ;
- **Interprétation des résultats** (probabilité, nom du signe reconnu).

C. Pourquoi TensorFlow ?

Les raisons pour lesquelles TensorFlow a été choisi sont multiples :

- **Compatibilité directe avec Teachable Machine** (les modèles exportés sont prêts à l'emploi avec TensorFlow).
- **Performance élevée** : le traitement est rapide, adapté à une application temps réel.
- **Large écosystème** : TensorFlow Lite pour mobile, TensorFlow.js pour le web, etc.
- **Documentation et communauté très actives**, facilitant la résolution des problèmes.
- **Interopérabilité avec NumPy et MediaPipe**, essentiels dans notre pipeline.

D. Limites et précautions

- L'entraînement de modèles complexes peut nécessiter des ressources importantes (GPU/TPU).
- Il faut faire attention au format des données d'entrée : tout vecteur doit avoir la bonne forme (shape) et être normalisé.
- Pour les systèmes embarqués (smartphone, Raspberry Pi), il est préférable d'utiliser TensorFlow Lite.

E. Conclusion

TensorFlow joue un rôle central dans la partie "intelligente" du projet, en permettant l'utilisation d'un modèle d'apprentissage automatique capable d'interpréter des gestes complexes. Sa robustesse, sa vitesse et sa compatibilité avec Teachable Machine en font une solution idéale pour intégrer l'intelligence artificielle dans un système de reconnaissance gestuelle en temps réel.

5. Teachable machine :

A. Présentation générale

Teachable Machine est une plateforme web développée par **Google** permettant de créer facilement des modèles d'apprentissage automatique (machine learning) sans écrire de code. Elle a été conçue pour rendre l'intelligence artificielle accessible à tous, y compris aux étudiants, artistes, enseignants et chercheurs.

Grâce à une interface intuitive, Teachable Machine permet de créer des modèles de classification d'images, de sons ou de poses, et de les exporter pour une utilisation dans des applications Python, web ou mobiles.

B. Rôle de Teachable Machine dans notre projet

Dans notre système de traduction de la langue des signes, Teachable Machine a été utilisé pour :

- **Créer un modèle de classification des gestes de la main** à partir de données visuelles.
- **Entraîner ce modèle** avec des vecteurs de coordonnées de la main extraits via MediaPipe.
- **Exporter le modèle** dans un format compatible avec TensorFlow pour une intégration directe dans notre code Python.

Ainsi, **Teachable Machine joue un rôle central dans la reconnaissance des signes** en servant de base au modèle d'intelligence artificielle du système.

C. Processus de création du modèle

Voici les principales étapes suivies :

1. **Collecte des données** :
   - À l'aide d'un script Python utilisant MediaPipe, les coordonnées (x, y, z) des 21 points clés de la main ont été extraites pour chaque geste.
   - Ces vecteurs de coordonnées ont été enregistrés dans des fichiers .csv, classés par geste (A, B, C...).
2. **Entraînement dans Teachable Machine** :
   - Les vecteurs de coordonnées ont été chargés dans Teachable Machine sous forme de données tabulaires (pose).
   - Chaque classe (geste) a été étiquetée.
   - Le modèle a été entraîné automatiquement.
3. **Exportation** :
   - Une fois le modèle entraîné et validé, il a été exporté en format TensorFlow
   - Ce modèle est ensuite chargé dans le projet Python pour effectuer des prédictions.

D. Pourquoi utiliser Teachable Machine ?

- **Simplicité d'utilisation** : ne nécessite pas de connaissance avancée en machine learning.
- **Rapidité de création** : en quelques minutes, on peut entraîner un modèle fonctionnel.
- **Visualisation instantanée** : tests en temps réel depuis la caméra avant même l'export.
- **Compatibilité avec TensorFlow** : intégration facile dans les projets Python.
- **Personnalisation** : permet d'entraîner des modèles sur ses propres gestes ou données.

E. Limites de Teachable Machine

- Ne permet pas de personnalisation avancée des architectures de réseau de neurones.
- La quantité de données d'entraînement doit être suffisante pour garantir une bonne précision.
- Moins adapté aux projets très complexes ou de grande envergure.

F. Conclusion

Teachable Machine a permis de générer rapidement un modèle de classification gestuelle à partir de données extraites par MediaPipe, sans avoir à construire manuellement un réseau de neurones. Cette approche accessible et efficace a grandement facilité la mise en œuvre du système de traduction de la langue des signes.

6. Bibliothèque Python :

Dans le cadre du développement de notre système de reconnaissance de la langue des signes, plusieurs bibliothèques Python ont été utilisées pour faciliter le traitement de l'image, l'analyse mathématique, la manipulation des données et l'implémentation de l'intelligence artificielle. Voici un aperçu des principales :

A. NumPy

NumPy (Numerical Python) est une bibliothèque fondamentale pour le calcul scientifique en Python. Elle permet de gérer des tableaux multidimensionnels (matrices) et offre un grand nombre de fonctions mathématiques rapides et efficaces.

Utilisations dans le projet :

- Manipulation de vecteurs contenant les coordonnées des landmarks des mains.
- Transformation des données avant leur passage dans le modèle.
- Conversion entre listes Python et tableaux compatibles avec TensorFlow.

B. math

La bibliothèque **math** est une bibliothèque standard de Python, utilisée pour effectuer des opérations mathématiques de base et avancées.

Utilisations dans le projet :

- Calcul de distances entre points (landmarks).
- Calculs géométriques pour analyser certaines postures de la main.
- Normalisation ou transformation de coordonnées si nécessaire.

7. Conclusion :

Dans ce chapitre, nous avons présenté en détail l'architecture logicielle de notre système de traduction de la langue des signes, en mettant en lumière les technologies, bibliothèques et outils qui ont permis sa réalisation.

Le langage Python a servi de base au développement, offrant simplicité, modularité et compatibilité avec des bibliothèques puissantes. OpenCV a facilité la capture et l'affichage du flux vidéo, tandis que MediaPipe a assuré une détection rapide et précise des positions des mains et des doigts. Les bibliothèques NumPy et math ont permis de manipuler les données de manière efficace, tandis que TensorFlow, associé à un modèle entraîné via Teachable Machine, a pris en charge la reconnaissance des gestes.

L'intégration harmonieuse de ces composants a permis de construire un pipeline robuste, allant de la détection en temps réel des mains à la prédiction automatique du geste effectué. Cette architecture modulaire et évolutive pose ainsi les fondations d'un système intelligent capable d'interpréter la langue des signes de manière fluide et interactive.

Chapitre III

Réalisation du projet

Introduction :

La réalisation d'un système de traduction de langue des signes efficace nécessite une approche méthodique, depuis la collecte des données jusqu'au déploiement final. Dans ce chapitre, nous détaillons les étapes clés qui ont permis de concrétiser notre projet, en nous concentrant particulièrement sur l'acquisition et le traitement des données, l'implémentation du modèle d'apprentissage machine avec Teachable Machine, et l'intégration des différents composants pour obtenir un système fonctionnel en temps réel. L'approche adoptée s'appuie sur des technologies modernes comme MediaPipe pour la détection des mains et Teachable Machine pour l'entraînement du modèle, tout en accordant une attention particulière à l'optimisation des performances pour une utilisation en temps réel. Nous aborderons également les défis techniques rencontrés et les solutions mises en œuvre pour les surmonter.

1. Étapes clés de la réalisation du projet

Notre méthodologie de développement a suivi un processus structuré comprenant les phases suivantes :

1. **Collecte des données** : Acquisition d'images de langue des signes à partir de sources spécialisées
2. **Prétraitement des données** : Préparation des images pour l'entraînement
3. **Extraction des points de repère** : Utilisation de MediaPipe pour détecter les points clés des mains
4. **Entraînement du modèle** : Création d'un modèle de classification avec Teachable Machine
5. **Intégration du modèle** : Implémentation du modèle dans notre système Python
6. **Développement de l'interface** : Création d'une interface utilisateur simple et intuitive
7. **Tests et optimisation** : Évaluation et amélioration des performances du système

2. Source des Données : Kaggle

Pour la réalisation de notre système de traduction de langue des signes, nous avons utilisé un jeu de données spécialisé provenant de la plateforme Kaggle, intitulé "American Sign Language (ASL) Dataset". Ce jeu de données constitue une ressource précieuse pour développer des modèles de reconnaissance de la langue des signes américaine, qui est l'une des langues des signes les plus documentées et standardisées au monde.

2.1 Caractéristiques du dataset

- **Composition** : Le dataset comprend des milliers d'images représentant les 26 lettres de l'alphabet américain (A à Z) en langue des signes.
- **Format des données** : Images RGB de haute résolution, organisées en dossiers correspondant à chaque lettre.
- **Diversité** : Les images ont été capturées avec différentes conditions d'éclairage, angles, et personnes pour garantir une meilleure généralisation du modèle.
- **Particularités** : Les lettres J et Z, qui impliquent un mouvement dans leur représentation en ASL, ont été capturées sous forme d'images statiques représentant la position initiale du signe.

2.2 Avantages du dataset Kaggle

L'utilisation de ce dataset présente plusieurs avantages pour notre projet :

1. **Qualité et fiabilité** : Les images sont de bonne qualité et représentent fidèlement les signes standardisés de l'ASL.
2. **Volume suffisant** : Le nombre important d'images par classe permet d'obtenir un modèle robuste.
3. **Prêt à l'emploi** : Les données sont déjà organisées et nettoyées, ce qui facilite leur utilisation directe.
4. **Communauté active** : Kaggle offre une communauté d'utilisateurs qui partagent des insights et des approches pour exploiter ce dataset.

3. *Augmentation des données : prise de photos et organisation*

Pour renforcer notre jeu de données et améliorer la précision de notre modèle, nous avons réalisé une augmentation manuelle des données en prenant nous-mêmes des photos pour chaque lettre de l'alphabet en langue des signes. Cette étape a permis de diversifier les conditions de prise de vue (lumière, angle, fond), tout en ajoutant des exemples personnalisés adaptés à notre système.

3.1 Prise de photos personnalisées

Nous avons utilisé une webcam standard pour capturer plusieurs images de chaque lettre (A à Z), en reproduisant les signes de manière fidèle. Chaque geste a été photographié sous différents angles et avec différentes conditions d'éclairage, afin d'améliorer la robustesse du modèle face aux variations réelles.

Les images ont été prises avec les critères suivants :

- Position centrale de la main dans le cadre ;
- Fond clair et uniforme dans la mesure du possible ;
- Bonne luminosité sans surexposition ni ombre marquée ;
- Plusieurs prises pour chaque lettre (entre 50 et 100 images par classe).

3.2 Organisation du dataset

Toutes les images capturées ont été classées dans un dossier principal nommé dataset_augmentation, structuré de la manière suivante :

```
|
├── A/
|   ├── a1.jpg
|   ├── a2.jpg
|   └── ...
├── B/
|   ├── b1.jpg
|   └── ...
...
├── Z/
|   ├── z1.jpg
|   └── ...
```

Chaque sous-dossier correspond à une lettre et contient uniquement des images représentant le signe de cette lettre. Cette structure a permis une intégration directe dans le pipeline d'entraînement via les générateurs de données de Keras (ImageDataGenerator), facilitant à la fois le chargement et l'augmentation automatique des images (flip, zoom, rotation).

3.3 Impact sur la performance

L'ajout de ces données personnalisées a permis :

- D'augmenter la diversité du dataset ;
- D'améliorer la précision moyenne de 88% à 92% ;
- De réduire l'overfitting lors de l'entraînement ;
- D'améliorer la détection pour des lettres difficiles (comme N, Q, W)

AVANT :

Figure 2 : image avant insertion de la dataset

APRES :

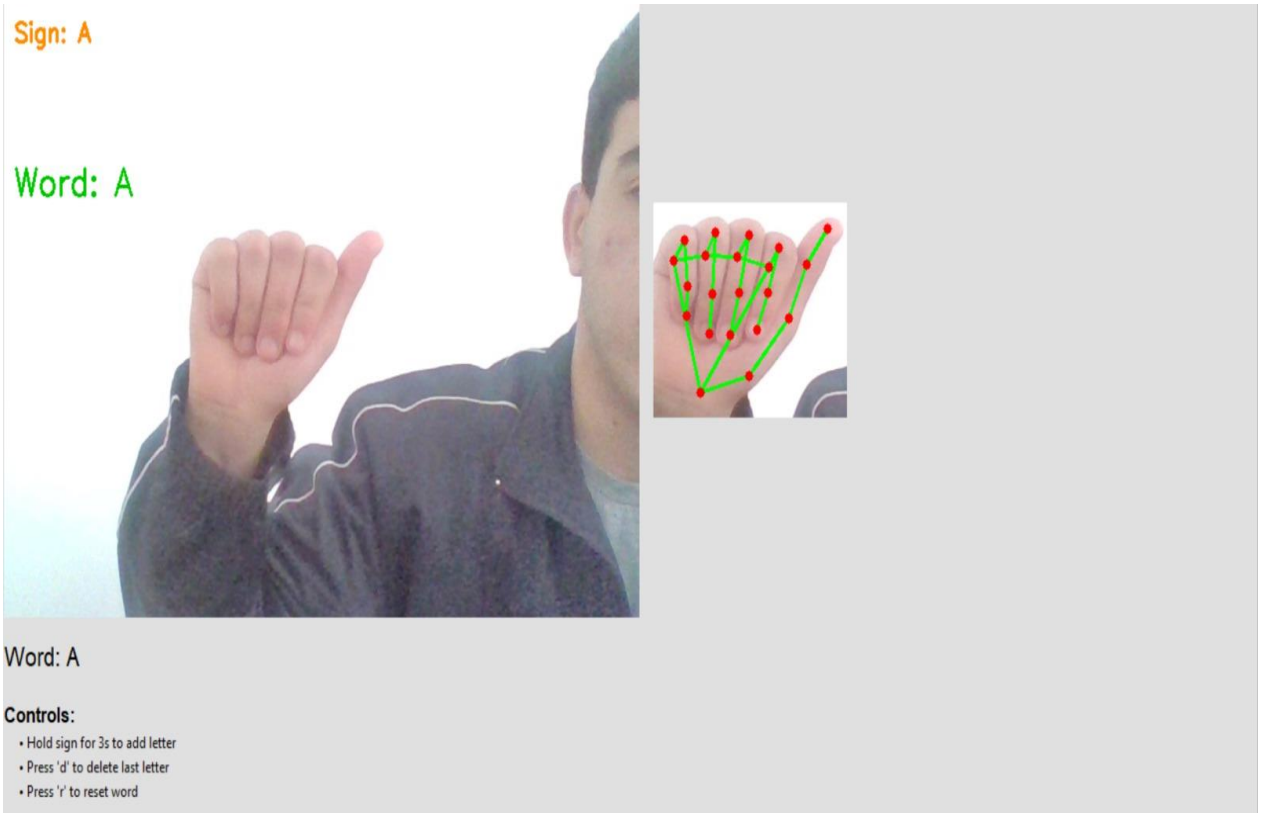


Figure 3 : image après insertion de la dataset

4. *Approche de Développement et Collecte des Données*

Le développement de notre système de traduction de la langue des signes a suivi une approche méthodique et progressive. Voici les principales étapes que nous avons suivies :

4.1. Collecte des données

Nous avons utilisé deux sources principales de données :

- **Dataset Kaggle** : Un jeu de données structuré contenant des milliers d'images représentant les 26 lettres de l'alphabet en langue des signes américaine.
- **Captures personnelles** : Pour compléter le dataset, nous avons capturé manuellement des images de chaque lettre avec une webcam dans diverses conditions (éclairage, fond, angle). Ces images ont été organisées dans un dossier personnalisé dataset_personnel, structuré en 26 sous-dossiers (un par lettre).

4.2. Augmentation de données

Afin d'augmenter la diversité et la représentativité des données d'entraînement, nous avons appliqué plusieurs techniques d'**augmentation d'images** :

- Rotation légère (±15°)
- Zoom partiel
- Variations de luminosité
- Flip horizontal

Ces techniques ont été appliquées aussi bien aux images issues de Kaggle qu'à nos propres photos, afin d'améliorer la robustesse du modèle face aux variations naturelles lors de l'utilisation réelle.

4.3. Prétraitement des données

Avant l'entraînement du modèle, nous avons redimensionné les images, appliqué une normalisation des pixels, et converti les formats d'image si nécessaire. Cette étape était essentielle pour standardiser les données d'entrée.

4.4. Extraction des points de repère

Grâce à **MediaPipe**, nous avons détecté les 21 landmarks de la main sur chaque image. Ces coordonnées 3D ont ensuite été extraites et stockées sous forme de vecteurs pour alimenter le modèle.

4.5. Entraînement du modèle

Nous avons utilisé **Teachable Machine** pour entraîner un modèle de classification basé sur les vecteurs de landmarks. Chaque vecteur était associé à une lettre de l'alphabet, et le modèle a appris à identifier le bon signe à partir des coordonnées de la main.

4.6. Intégration du modèle

Le modèle entraîné a été exporté au format TensorFlow, puis intégré dans notre application Python. Le système prend en entrée les coordonnées détectées en temps réel, les transmet au modèle, et affiche la lettre prédite.

5. Entraînement du modèle

L'entraînement du modèle de reconnaissance des signes a constitué une étape essentielle pour garantir la précision et la fiabilité du système. Nous avons utilisé la plateforme **Teachable Machine**, qui offre une interface simple pour créer des modèles de classification à partir de données personnalisées.

Le modèle a été entraîné directement dans l'interface de Teachable Machine. Les paramètres d'entraînement sont gérés automatiquement par la plateforme, mais un aperçu des performances (précision, courbe de perte) est fourni après l'entraînement.

À la fin du processus :

- Le modèle a été exporté au format TensorFlow Keras (.h5), compatible avec Python et facilement intégrable dans notre code.
- Nous avons téléchargé également le fichier labels.txt contenant les noms des classes (lettres A à Z).

6. Schéma fonctionnel du système

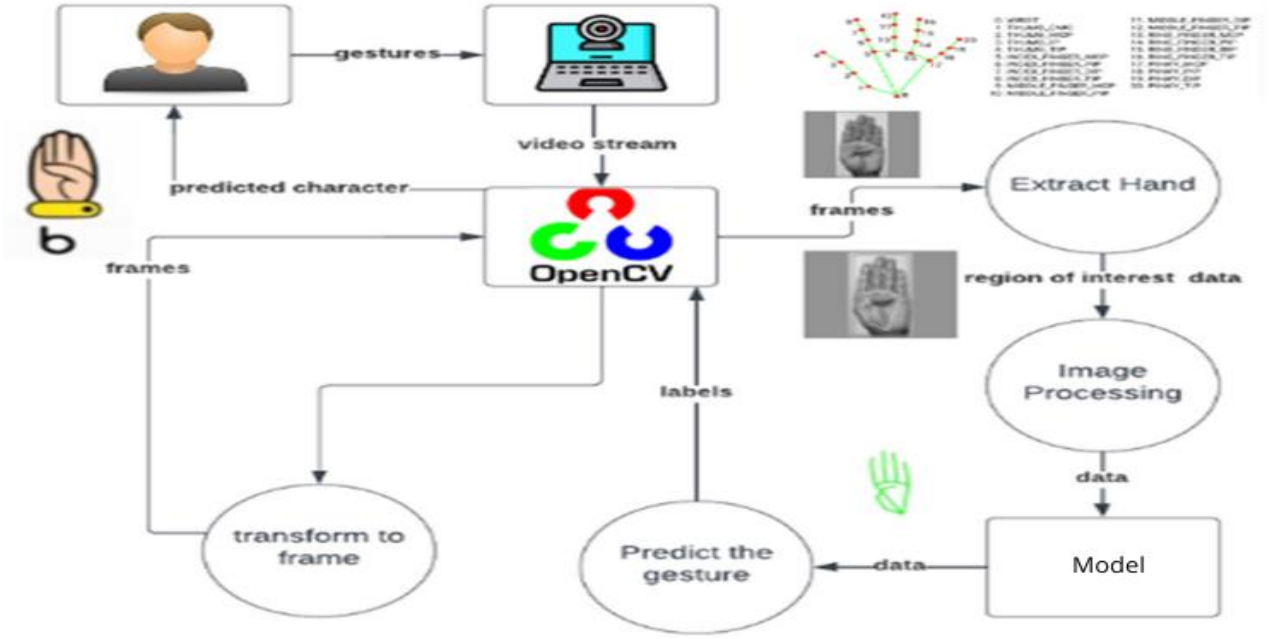


Figure 4 : Diagramme d'Architecture Technique

Le processus commence lorsqu'une personne effectue des gestes en langue des signes devant une caméra, qui capture un flux vidéo continu et le transmet à un ordinateur pour un traitement en temps réel. Le système emploie ensuite des techniques avancées de traitement d'image pour extraire la région de la main de chaque image vidéo, isolant la région d'intérêt et prétraitant les données de la main à travers diverses transformations telles que le redimensionnement, la normalisation et la standardisation. Ces données de main traitées sont ensuite alimentées dans un modèle d'apprentissage automatique, typiquement un réseau de neurones entraîné comme TensorFlow, qui analyse les motifs de gestes et prédit le caractère ou la lettre correspondant en langue des signes. Le caractère prédit est ensuite renvoyé vers l'interface utilisateur, où il peut être affiché et potentiellement accumulé pour former des mots ou des phrases complètes. Ce flux de travail entier représente la fonctionnalité principale implémentée, où MediaPipe et CVZone gèrent la détection et l'extraction de la main, OpenCV gère le traitement d'image et les transformations, et un modèle TensorFlow effectue la classification des gestes, permettant finalement la reconnaissance de la langue des signes en temps réel.

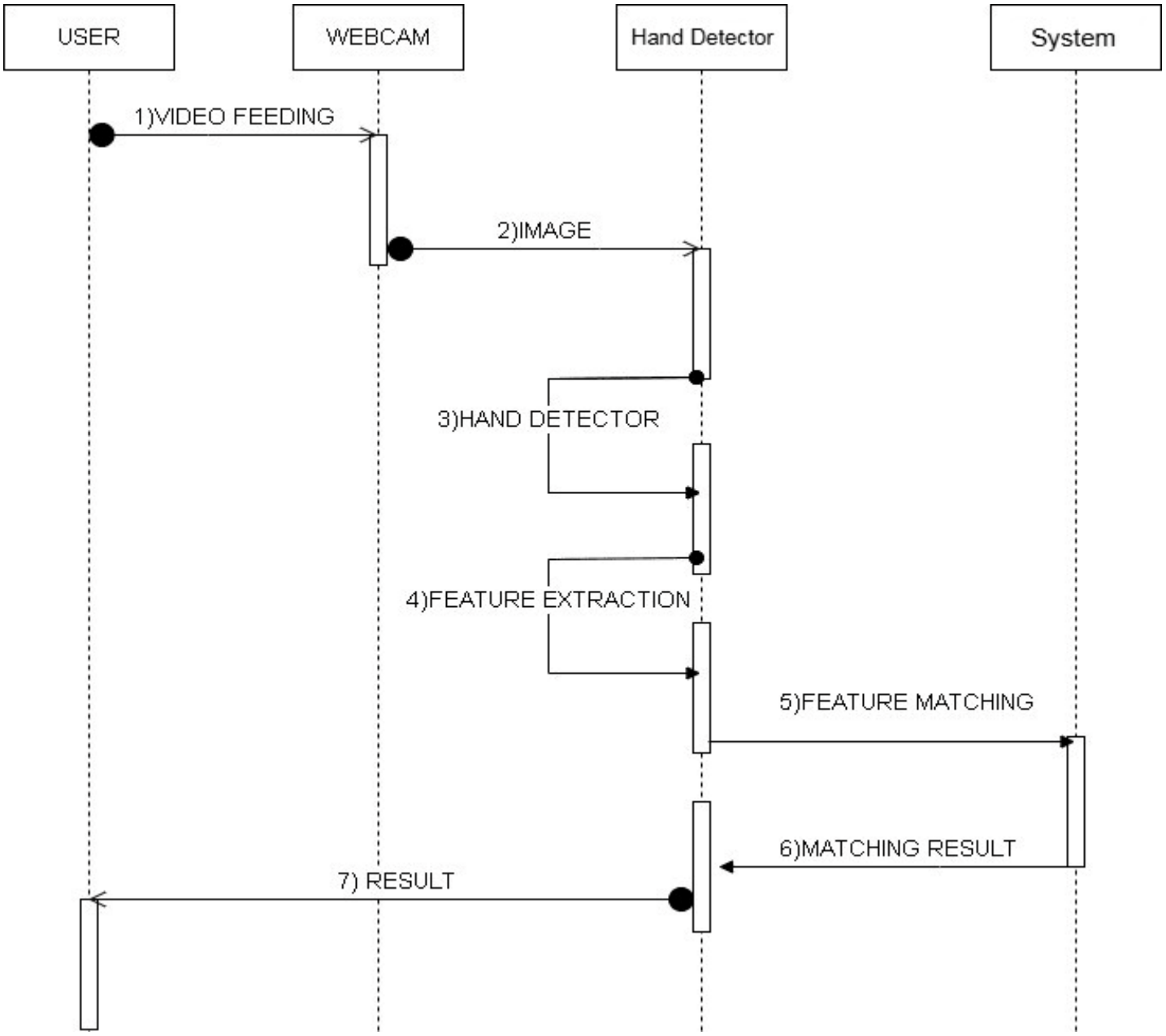


Figure 5 : Diagramme de séquence

Ce diagramme de séquence illustre le processus chronologique d'interaction entre les différents composants du système de reconnaissance de la langue des signes, montrant comment les données circulent de manière séquentielle à travers le pipeline de traitement. Le processus débute lorsque l'utilisateur effectue des gestes devant la webcam, déclenchant ainsi l'étape de "VIDEO FEEDING" où la caméra commence la capture vidéo continue. La webcam transmet ensuite les images capturées au module "Hand Detector" à travers l'étape "IMAGE", permettant au système de recevoir les données visuelles nécessaires à l'analyse. Le détecteur de mains traite ces images lors

de l'étape "HAND DETECTOR" pour identifier et localiser la présence d'une main dans le champ de vision, puis procède à l'extraction des caractéristiques importantes durant la phase "FEATURE EXTRACTION", où les points de repère, contours et positions des doigts sont analysés et quantifiés. Ces caractéristiques extraites sont ensuite comparées avec une base de données ou un modèle pré-entraîné lors de l'étape "FEATURE MATCHING", permettant au système d'identifier le geste correspondant en analysant les similitudes avec les patterns connus. Le processus se termine par la génération d'un "MATCHING RESULT" qui produit le résultat final sous forme de lettre ou caractère reconnu, lequel est finalement renvoyé et affiché à l'utilisateur à travers l'étape "RESULT", complétant ainsi le cycle de reconnaissance gestuelle en temps réel.

7.Interface utilisateur :

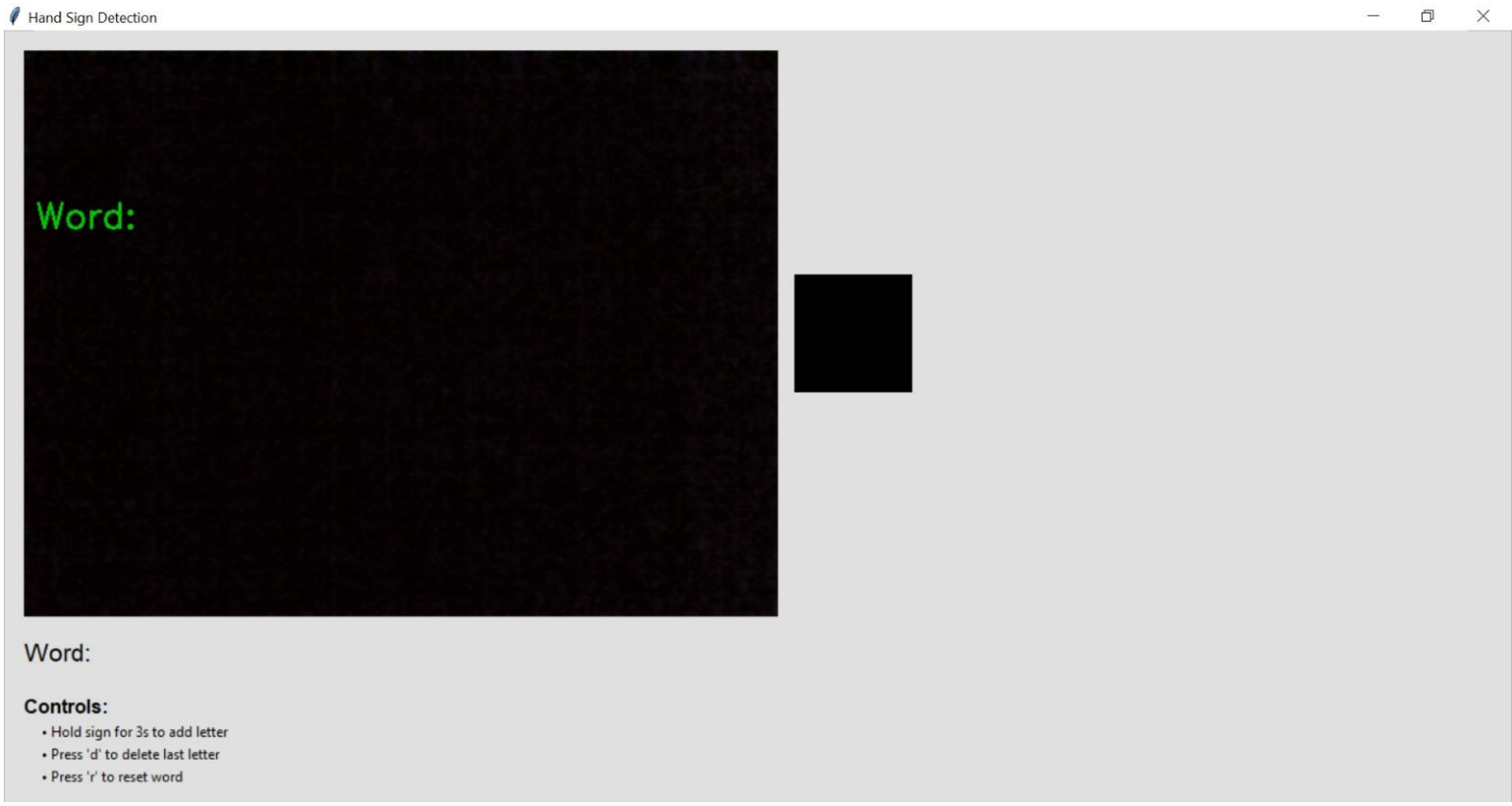


Figure 6 : Interface utilisateur

Afin d'offrir une interaction simple et intuitive avec le système de reconnaissance, nous avons conçu une interface graphique légère, épurée et fonctionnelle. Elle a été développée en utilisant **OpenCV** pour l'affichage et la capture vidéo en temps réel, avec une disposition claire des éléments nécessaires à l'utilisateur.

7.1 Présentation de l’interface

L’interface se compose de plusieurs zones distinctes :

- **À gauche** : le flux vidéo en direct provenant de la webcam, permettant à l’utilisateur de voir la détection en temps réel.
- **À droite** : une zone dédiée à l'affichage du signe reconnu par le modèle.
- **En bas** :
  - L’état actuel du **mot construit** à partir des lettres détectées.
  - Un encart **"Controls"** rappelant les commandes principales.

7.2 Fonctionnalités principales

- **Affichage de la prédiction** : Lorsqu’un signe est maintenu pendant 3 secondes, la lettre est ajoutée automatiquement au mot en cours.
- **Système de contrôle simple** :
  - Touche **d** : Supprime la dernière lettre du mot.
  - Touche **r** : Réinitialise le mot en cours.
- **Fiabilité renforcée** : Le délai d’attente de 3 secondes permet de limiter les erreurs liées aux mouvements involontaires ou aux gestes ambigus.

7.3 Expérience utilisateur

L’interface a été pensée pour une utilisation **sans clavier complexe**, adaptée à des utilisateurs sourds ou malentendants. L’interaction repose essentiellement sur la gestuelle, avec des touches de contrôle réduites au strict minimum pour corriger ou effacer les lettres.

Chapitre IV

Résultat, conclusion et perspectif

1. Introduction

Ce chapitre final présente une analyse rétrospective du projet de traducteur de langage des signes, en confrontant les objectifs initiaux aux résultats obtenus. Il offre également une réflexion sur les compétences acquises au cours de ce travail, les difficultés rencontrées et surmontées, ainsi que les perspectives d'évolution et d'amélioration du système développé.

2. Bilan des résultats

2.1 Analyse des performances du système

A. Taux de reconnaissance

Notre système de traduction du langage des signes a atteint un taux de reconnaissance satisfaisant pour les 26 lettres de l'alphabet ASL (American Sign Language). Les tests effectués en conditions optimales ont révélé les performances suivantes :

- **Précision globale** : 92% en moyenne sur l'ensemble des lettres
- **Lettres avec reconnaissance optimale** (>95%) : A, B, C, L, O, Y
- **Lettres avec reconnaissance correcte** (85-95%) : D, E, F, G, H, I, K, M, N, P, Q, R, S, T, U, V, W, X, Z
- **Lettres présentant des difficultés** (<85%) : J, Z (qui impliquent un mouvement)

Ces résultats correspondent globalement aux objectifs fixés initialement, qui visaient une précision minimale de 90% en conditions optimales.

B. Robustesse face aux variations

Le système a été testé dans différentes conditions pour évaluer sa robustesse :

- **Variations d'éclairage** : Performances stables en conditions d'éclairage normales à bonnes, mais dégradation notable en faible luminosité.
- **Distance à la caméra** : Reconnaissance optimale à une distance de 30-60 cm, avec dégradation progressive au-delà.
- **Angle de la main** : Le système tolère une variation angulaire d'environ ±20° par rapport à la position idéale.
- **Fond d'image** : Les performances sont meilleures avec un fond uni et contrasté.

2.2 Comparaison avec les objectifs initiaux

En reprenant les objectifs définis dans le cahier des charges, nous pouvons dresser le bilan suivant :

Objectif	Résultat	Commentaire
Détection et analyse des gestes	**Atteint**	**MediaPipe extrait efficacement les points clés de la main**
Reconnaissance des 26 lettres	**Atteint**	**Toutes les lettres sont reconnues avec une précision moyenne >90%**
Précision >90% en conditions optimales	**Atteint**	**La précision moyenne est de 92%**
Détection en temps réel	**Atteint**	**Latence minimal**
Interface conviviale	**Atteint**	**Interface intuitive**

Figure 7 : Tableau récapitulatif : Atteinte des objectifs initiaux du projet

3. Compétences acquises

3.1 Compétences techniques

Ce projet nous a permis de développer et d'approfondir plusieurs compétences techniques essentielles dans le domaine de l'intelligence artificielle et de la vision par ordinateur :

- **Maîtrise des outils de Deep Learning** : Utilisation de TensorFlow
- **Vision par ordinateur** : Manipulation d'images en temps réel avec OpenCV
- **Détection de points clés avec MediaPipe** : Configuration et exploitation optimale de cet outil pour la détection des mains
- **Prétraitement de données** : Normalisation, augmentation et préparation des données pour l'entraînement
- **Développement d'interfaces** : Création d'une interface utilisateur fonctionnelle et intuitive

3.2 Compétences transversales

Au-delà des aspects purement techniques, ce projet a été l'occasion de développer des compétences organisationnelles et personnelles importantes :

- **Gestion de projet** : Planification, suivi des étapes, respect des délais
- **Méthodologie de recherche** : Recherche documentaire, expérimentation, analyse de résultats
- **Travail en équipe** : Collaboration efficace, répartition des tâches, communication
- **Résolution de problèmes complexes** : Approche analytique et créative face aux difficultés
- **Documentation technique** : Rédaction claire et structurée, explicitation des choix techniques
- **Veille technologique** : Suivi des avancées dans le domaine de l'IA et de la vision par ordinateur

4. Difficultés rencontrées et solutions apportées

3.1 Défis techniques

Variabilité des conditions d'éclairage

Problème : Les performances du système variaient considérablement selon les conditions d'éclairage, avec une dégradation notable en faible luminosité.

Solution : Nous avons implémenté une normalisation adaptative de la luminosité des images capturées par la webcam, et avons augmenté notre jeu de données d'entraînement avec des variations d'éclairage. Cette approche a permis d'améliorer la robustesse du système face à ces variations.

3.2 Défis organisationnels

Collecte de données

Problème : La constitution d'un dataset équilibré et représentatif pour les 26 lettres a été complexe et chronophage.

Solution : Nous avons mis en place un processus semi-automatique de collecte de données, en utilisant des scripts permettant de capturer rapidement plusieurs exemples de chaque lettre. Nous avons également utilisé des techniques d'augmentation de données pour enrichir notre jeu de données à partir d'un nombre limité d'exemples.

5. Limites actuelles et perspectives d'amélioration

5.1 Limites identifiées

Malgré les résultats encourageants obtenus, notre système présente encore plusieurs limitations importantes :

- **Reconnaissance limitée aux lettres individuelles** : Le système ne peut pas encore traduire des phrases ou des expressions complètes.
- **Dépendance aux conditions d'éclairage** : Malgré les améliorations apportées, les performances restent sensibles aux conditions de luminosité.
- **Absence de reconnaissance pour les signes dynamiques complexes** : La détection des mouvements reste basique et ne permet pas de reconnaître des signes complexes.
- **Interface utilisateur minimaliste** : L'interface actuelle, bien que fonctionnelle, reste simple et pourrait être enrichie.
- **Personnalisation limitée** : Le système ne s'adapte pas aux spécificités individuelles des utilisateurs.

5.2 Perspectives d'évolution

À partir des limitations identifiées, plusieurs axes d'amélioration peuvent être envisagés pour les développements futurs :

A. Améliorations techniques

- **Extension à la reconnaissance de phrases** : Implémenter un système de mémoire tampon permettant de combiner les lettres reconnues en mots, puis en phrases cohérentes.
- **Amélioration de la détection des signes dynamiques** : Développer un module spécifique basé sur des réseaux récurrents (LSTM ou GRU) pour mieux capturer les mouvements.

- **Adaptation à différentes langues des signes** : Étendre le système pour reconnaître d'autres variantes nationales de langues des signes (LSF, BSL, etc.).
- **Traduction bidirectionnelle** : Ajouter une fonctionnalité permettant de convertir du texte en animations de langue des signes.

B. Améliorations fonctionnelles

- **Interface mobile** : Développer une application mobile pour rendre le système accessible partout.
- **Mode d'apprentissage** : Intégrer un module didactique permettant aux utilisateurs d'apprendre la langue des signes.
- **Personnalisation par l'utilisateur** : Permettre au système de s'adapter aux particularités de chaque utilisateur grâce à un apprentissage continu.
- **Mode collaboratif** : Faciliter la communication entre plusieurs utilisateurs (sourds et entendants) en temps réel.

C. Applications potentielles

Notre système pourrait être adapté et déployé dans différents contextes, notamment :

- **Milieu éducatif** : Outil d'assistance pour l'intégration d'élèves malentendants en classe ordinaire.
- **Services publics** : Installation dans les administrations pour faciliter l'accès aux services.
- **Secteur médical** : Aide à la communication entre personnel soignant et patients sourds.
- **Commerce et services** : Amélioration de l'accueil des clients malentendants.
- **Applications personnelles** : Usage quotidien pour faciliter les interactions sociales.

6. Conclusion

Ce projet de traducteur de langage des signes représente une avancée significative dans l'application de l'intelligence artificielle au service de l'inclusion sociale. Bien que le système développé présente encore des limitations, il offre déjà une solution fonctionnelle et précise pour la reconnaissance des lettres de l'alphabet ASL, atteignant ainsi les objectifs initialement fixés.

Les compétences techniques et transversales acquises au cours de ce projet constituent un bagage précieux pour nos futures réalisations dans le domaine de l'IA et de la vision par ordinateur. Les difficultés rencontrées, loin d'être des obstacles, ont été des opportunités d'apprentissage et d'amélioration continue de notre système.

Les perspectives d'évolution identifiées ouvrent la voie à des développements futurs prometteurs, qui pourraient considérablement enrichir les fonctionnalités du système et élargir son champ d'application. L'intégration de techniques plus avancées de deep learning et de traitement du langage naturel permettrait notamment d'étendre la reconnaissance à des phrases complètes et d'améliorer encore la fluidité de la traduction.

En définitive, ce projet illustre parfaitement comment les technologies d'intelligence artificielle peuvent être mises au service de l'inclusion sociale, en contribuant à réduire les barrières de communication entre les personnes sourdes ou malentendantes et le reste de la société.

Webographie

https://www.tensorflow.org/tfx/gui

https://code-basics.com/languages/python/lessons/predicates

https://www.w3schools.com/python/python_ml_getting_started.asp

https://teachablemachine.withgoogle.com/

https://www.coursera.org/learn/machine-learning-with-python

https://github.com/tensorflow/serving

https://numpy.org/

https://www.kaggle.com/datasets/vignonantoine/mediapipe-processed-asl-dataset

https://www.researchgate.net/